SPECIFICATION

Docket No. 0544MH-40014

TO ALL WHOM IT MAY CONCERN:

BE IT KNOWN that we, Daniel Brown, William Lance Eason, and George A. Thompson, residing in the State of Texas, have invented new and useful improvements in a

RULES-BASED NOTIFICATION SYSTEM

of which the following is a specification:

CROSS-REFERENCE TO RELATED APPLICATION

This application claims the benefit of United States Provisional Application No. 60/158,732, filed October 11, 1999. This application also contains matter in common with copending US application Serial No. 09/686,442, filed on even date herewith, titled CUSTOMIZABLE USER WINDOW, which is incorporated by reference.

DTC 4/21/03

BACKGROUND OF THE INVENTION

1. Field of the Invention:

The present invention relates generally to communications for electronic computer systems, and more specifically to a system and method for notifying a user that selected events have occurred within the system.

2. Description of the Prior Art:

As computer systems and data communications become increasingly widespread, more and more users are able to perform a high percentage of their daily tasks over various communication networks. For example, many users are able to obtain key information in a timely manner through the use of interconnected software and hardware systems.

Because of the great wealth of information available, it often becomes time consuming for a user to keep up with items and events of interest. In some cases, a dedicated window may be included on a user's desktop to provide information about high priority items. For example, a streaming stock quote ticker may be resident on a user's desktop to enable them to keep up with selected stock market information. However, the number and variety of sources of potentially interesting information is so large that this type of dedicated approach can work for only a small number of sources.

Often, a user is required to access a database or other source of information in order to determine whether an event of interest has occurred. This can be relatively time consuming, and often results in undesirable delays if the user is unable to check the required sources often enough.

Another approach is to automatically send notifications directly from an application to a user when selected events occur. This requires coding of the necessary handling and routing modules within the application. This can become a significant burden from the application writer's standpoint, and can lead to inconsistencies in the way message notifications are presented to users.

It would be desirable to provide a system and method for automatically notifying users upon the occurrence of selected events. It would further be desirable for such notification system to be able to handle different types of events and present notifications to users in different ways.

SUMMARY OF THE INVENTION

In accordance with the present invention, a system and method for providing notifications to a user includes an alert manager. The alert manager receives notifications of events incurring within various executing applications, and utilizes rules to determine when alert notifications should be sent to one or more specified users. In addition, operation of batch jobs can determine changes in state of selected data, which generate events for the alert manager in addition to those generated directly by an executing application. Alert notifications to users can be sent to the user via any of several communications channels.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself however, as well as a preferred mode of use, further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a high-level block diagram of a system in accordance with the present invention;

Figure 2 is a block diagram indicating additional details of the system of Figure 1;

Figures 3A and 3B illustrate operation of different types of event generation techniques;

Figure 4 shows the type of information maintained by an event router in accordance with the present invention;

Figure 5 is a block diagram illustrating an operation of an alert manager;

Figure 6A illustrates one preferred embodiment for using a rule filter with event handlers; and

Figure 6B illustrates an alternative embodiment for using a rule filter with event handlers.

DESCRIPTION OF THE PREFERRED EMBODIMENT

As will be understood by those skilled in the art, the present system and method can be used with nearly any computer system in which events occur during execution of an application, or in which data is changed. In the preferred embodiment, the operating applications are of a type which generate messages indicating the occurrence of selected events. However, as described below, the system will also work with applications which do not generate such events, and is adaptable to nearly any type of computer application.

Referring to Figure 1, a preferred embodiment of the inventive system and method is illustrated generally as reference number 10. A plurality of applications 12 operate concurrently, and may be located on a single piece of hardware, or spread among numerous machines. The applications 12 need not operate in conjunction with one another, but may be a wide variety of selected applications, of any type, which perform data processing functions.

As will be described in more detail in connection with Figure 2, applications 12 generate events 14, which are forwarded to an event router 16. Events 14 are basically messages indicating that some type of predetermined event has occurred. As defined further in connection with Figure 2, these messages may be generated directly by applications to indicate that an event has occurred in that application. For example, an application which is used to change the price of a product in a

database can generate an event message indicating the product number and new price for the product. In addition, as described below, certain types of applications 12 may periodically scan persistent information, such as that stored in databases, to determine whether data has changed. These applications can also generate events showing that changes of the selected type have occurred.

Event router 16 is described in more detail in connection with Figure 4, but is generally an application which receives incoming events 14, and routes them to recipients which have registered to receive events of this type. For example, if a user or another application has registered to receive notification of price change events, the event router 16 will maintain this information. When an incoming price change event is received by event router 16, it will then be forwarded to all applications which have been registered to receive this event.

Event router 16 therefore acts as a central routing point for messages. It is not necessary for applications 12 to know, or be able to find out, who will receive notifications of events. Instead, applications 12 simply generate defined events and send all of them to event router 16. Event router 16 handles the task of sending the events to as many recipients as are registered to receive them.

Some events 18 are routed to various types of event handlers 20, which then use the incoming event message to process data. In many cases, event handlers 20 will also be classified as applications 12. In this application, event

handlers 20 are generally any applications which make some use of the information provided by incoming event messages 18.

Some events 22 are routed to an alert manager 24, which is a particular type of event handler. As described further in connection with the remaining figures, alert manager 24 handles incoming events by determining whether any incoming event requires that a notification, or alert 26, to be forwarded to a user 28. Alert manager 24 utilizes a set of rules to determine to whom, and when, notifications should be made to a user 28.

Referring to Figure 2, the types of applications 12 which generate events may be classified generally into two categories. The first category is referred to herein as business objects 30, which may include within them state machines 32 and similar operating modules. Business objects 30 generate "explicit" events, meaning that the code of the business object application explicitly generates an event to be sent to event router 16 as events occur. This capability must be programmed into business objects 30 and their state machines 32.

Explicit events are useful, in the present invention, for indicating when a single event has occurred. For example, when a sale is made, or a product changes price, or a new product becomes available, if the corresponding business objects are properly programmed events will be generated. These can be picked up by alert manager 24, and used to generate notifications.

A second kind of application that can generate events is referred to herein generally as "batch jobs." These jobs are applications that, generally, periodically check persistent data, such as data stored in a database, and look for changes that may have occurred. For example, if an application which enters new products into a database is not one which has previously been coded as a business object, to generate explicit events on this occurrence, a batch job 34 can periodically scan a product database and determine when new products have been added. Events which are discovered by such a comparison between a previous state of an object, in a persistent memory, with the current state are referred to herein as "implicit events."

Use of batch jobs to scan data looking for implicit events is useful both for events which occur over time, and for use with applications which are not already coded to generate the desired explicit events.

Event router 16 can route events to two different types of handlers. These can be categorized generally as synchronous handlers 36 and asynchronous handlers 38. Events which are intended for use by asynchronous handlers are preferably placed into a queue 40, from which they are later withdrawn by various asynchronous handlers 38. More than one queue 40 can be provided, and registration of a handler with the event router, to be notified of a particular type of event, will indicate whether the handler is synchronous, or asynchronous, and in the later case indicate which queue is used for the event message.

Figures 3A and 3B illustrate the operational difference between explicit and implicit events. In Figure 3A, dataflow occurring upon an explicit event is shown. In this example, a user running an application enters a new product, number 789, and identified as a "BOAT", into the system.

When the user desires to add the product, procedure add-product is executed 42. This procedure passes the new product number, '789', and the new item description, 'BOAT', to business object 44. When the new product is set up by business object 44, an event 46 is generated and sent to event router 16. Event 46 indicates that a new product has been added, and passes the relevant parameters as part of the message.

Referring to Figure 3B, addition of the same product is illustrated when no business object is available to generate an explicit event. Instead, it will be assumed that an appropriate batch object 34 runs periodically to check a persistent data base. In this example, the batch job runs at 9:00 o'clock, and detects a table in the data base which shows two products, numbers 123 and 456. This information is retained by the batch object for future use. At a subsequent time, 9:08 in this example, a user executes the add-product procedure as was the case in Figure 3A. However, business object 50 is one which does not generate an explicit event message when the data base is updated. Thus, no message is sent to event router 16 at 9:08.

At 10:00, in this example, the batch job executes again. At this time, the information it collects is shown in table 52 and includes three products. By comparing the present table 52 with earlier table 48, the batch job determines that a new product has been added to the data base. The batch job then generates an implicit event showing that the new product has been added, and sends it to router 16.

Depending upon the implementation, it may be desirable for implicit events to carry a flag distinguishing them from explicit events to assist either the router or handler in determining how to treat the message. However, in other cases, it may not be necessary to distinguish between implicit and explicit events; only the fact that an event occurs would be of interest to the handlers.

Figure 4 indicates the type of information retained by event router 16 in order to route events which it receives. The information is shown in Figure 4 as a table, but may be stored internally within event router 16 in any desirable form.

Referring to Figure 4, the table indicates that event router 16 stores a list of types of events which are to be routed. This table is preferably dynamic, and can be added to as various handlers register with event router 16. In the example of Figure 4, EVENT TYPE 1 has three recipients, R1, R2 and R3, registered to receive a copy of this type of event. Thus, when an event of EVENT TYPE 1 is received by event router 16, recipients R1, R2 and R3 in turn receive a copy of the event message. Identification of the recipient indicates where the message will be

sent, and any other particular conditions under which it should be forwarded. If any recipients R1, R2 or R3 are synchronous handlers, the event message will be copied and forwarded to them immediately. If any of these recipients are synchronous events, the registration of such event will indicate a queue into which a copy of the event message should be placed.

Figure 4 shows five different event types, with various recipients registered to receive them. Some event types, such as EVENT TYPE 3 in this case, may not have any registered recipients. In such case, any events of this type are not routed to a handler. In most cases, however, one or more recipients will be registered to receive an event, and event router 16 will forward copies of the event message to all registered recipients.

By use of the event router, recipients can register to receive messages of various types without having any impact on the business objects which generate these events. Instead, the events are consistently passed to event router 16, from which any handler which needs to be copied on the event message receives its forwarded copy. Each handler will then deal with a message according to its internal logic.

Figure 5 is a block diagram illustrating functioning of the alert manager 24. As described above, alert manager 24 is preferably an asynchronous handler and therefore receives its events from a queue.

Within alert manager 24 are two major portions. These include a rules portion 60, and a notifications portion 62. Rules portions 60 contains a large number of rules defining when events are to be acted upon. Each user who desires to receive alert notifications from alert manager 24 will register with the alert manager, and define the conditions under which that user wishes to receive a notification. Rules are generally conditional statements which define where the notification is to be generated.

Sub A3 In the example show in Figure 5, two rules are provided. The conditional for the first rule provides that a product equals ("BOAT"), and a price less than $7,500.00. If an event occurs which makes this conditional TRUE, a corresponding notification 62 will be generated to the user requesting this alert.

In this example, a price-change event 64 is passed to alert manager 24 by its associated queue, the rules portion 60 compares the context data passed along with the message with its rule conditional portions. The comparisons are performed by a rules engine 66, which receives rules and context data from rules portion 62 and returns Boolean values of true or false depending upon the evaluation of the conditional. In this example, rules engine 66 returns a value of TRUE for the first rule, because the product is "BOAT" and the price is less than $7,500.00.

When the condition of a rule is satisfied, the corresponding notification is then executed and sent to the user 28. Notifications can involve sending a message by a computer communications network to a user's desktop. Notifications

can also include sending of a message via phone, pager, fax, or any other desired technique. The contents of the message are defined by the notification portion of the rule. Preferably, a fairly large number of preselected message types will be available, so that a user will be able to select from these messages. The messages can include textual information, and can include as parameters some or all of the context information included with event 64. In the example of Figure 5, an e-mail message can be sent to the user who registered Rule 1 indicating that a price change for BOAT has occurred, and indicating the new price for the BOAT. A call can be made to a pager at the same time; any desired number of notifications can be made.

This rule base system allows users to register with the alert manager 24, defining logical conditions under which alerts of various types will be sent. This frees the user from having to check for events or changed conditions individually; this is done automatically by the rules set up in the alert manager. Users can determine how these messages are to be sent. E-mail would be one typical type of message; users may also provide for one or more notification windows to be generated upon their desktop for the sole purpose of receiving alert notifications.

By setting up and registering different types of alerts with a central system, a user can be notified regarding a wide variety of events which would otherwise take too much time and effort to profitably be viewed. Upon receiving one of these alerts, the user can, if she so desires, take a corresponding action.

Sub A4 The method described above can be generalized and modified to provide different functionality. Referring to Figure 6A, a diagram is shown of a system which uses a rule filter similar to the rules portion of an alert manager 24 for event handlers in general. When business object 30 generates an event 14, it is communicated to event router 16. Event router 16 then sends copies of event 70, 72 to event handlers 74 and 76 which have been previously registered with event router 16. Event handlers 74, 76 each have a rule filter 78, 80, respectively, which uses conditionals in essentially the same manner as described in connection with Figure 5. These rule filters allow for initial filtering of events by event handler 74, 76 as part of the determination of which events to respond to.

Sub A5 An alternative approach is shown in Figure 6B, in which a combined router/filter includes event router 16 and rule filter 84. When business object 30 sends an event 14 to event router 16, rule filter 84 is checked to see whether any rules apply to such event. If a conditional of a rule is met, the functional portion of the rule, directing events to particular handlers, is then executed. For example, when event 14 is sent to event router 16, the corresponding copy is not automatically sent to both event handlers 86, 88. Instead, the context data included with event 14 is evaluated as described in connection with Figure 5 within rule filter 84, and events are only sent to handlers for which the rule conditional evaluates true. In this example, event copy 90 is forwarded only to event handler 88 because no corresponding conditional was satisfied which would cause event 14 to be routed to event handler 86.

Inclusion of rule filter 84 with event handler 16 allows for several changes in the system. Initially, router/filter 82 must be more complicated, and allow handlers to register rules with filter 84 as well as simply being added to a list in event router 16. Preferably, rules in filter 84 are checked only for handlers which are registered for a particular event type, so that only a few rules need to be checked when an incoming event is received.

Inclusion of rule filter 84 with event router 16 increases the complexity of the code associated with the event router, but decreases communications to event handlers. Only events which satisfy a predefined criteria are passed to an event handler by rule filter 84. In addition to lessening network traffic, this approach simplifies all of the event handlers. Each event handler may be simplified by deleting any initial rule filter used to handle incoming events; instead, this filtering is performed by rules registered with rule filter 84.

The system and method described above provide a simplified method for generating user alerts in response to various events taking place within applications. By sending all events to a central location, an alert manager can register for any desired type of event and, in response to logical conditionals being satisfied, send alert notifications to any registered users. Because the system can handle both explicit and implicit events, it is suitable for use with existing applications which do not generate events suitable for routing through event router 16, or which do not generate events of interest to a particular user.

Users gain the ability to select types of events for which they wish to be notified. When these events occur, notification occurs automatically. The user need not periodically review various locations to determine whether anything has happened. Users can select from among types of events, and the applications writers can determine which events to actually make available. For example, a user can register with the alert manager 24 to be notified whenever a new product is introduced by a competitor; automatic notifications will be provided whenever this occurs, assuming the events are available as either explicit or implicit events.

Preferably, users will be able to select from one or more menus of choices provided by the system. This simplifies the task of registering to receive alerts, and ensures a certain uniformity of notifications across all users. The conditionals used in the alert rules can be provided as templates, with the user selecting the form of the conditional and any particular values to be used. In this manner, the user can easily provide a set of alert rules to meet his or her needs.

As described previously, the alert notifications themselves can be provided in any available format supported by the system. Notification may be by e-mail or other electronic messaging as known in the art. By sending appropriate messages to any type of intermediate interface devices, messages such as pages or telephone alerts can also be made. Because the alert notification message and its type are maintained in tables in the alert manager, addition of a new technology is easily made to the alert system. All that is necessary is to provide that a selected message be sent to an appropriate handler form the alert manager, and the message can be sent to the registered user.

While the invention has been particularly shown and described with reference to a preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.